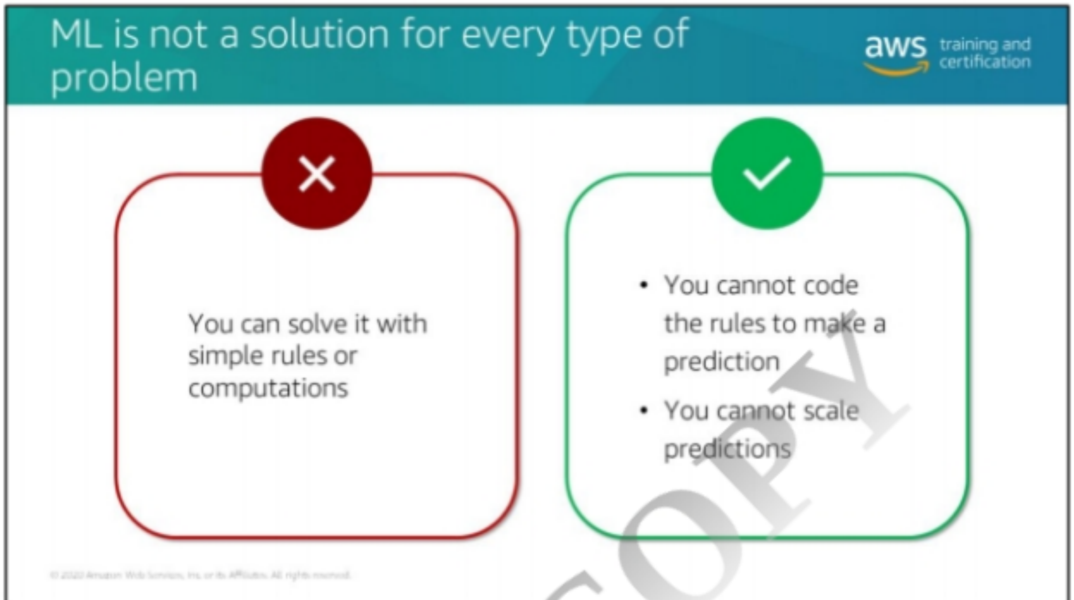


Now, think of a business case that is not appropriate for ML. ML is not appropriate solution when you can determine a target value by using simple rules or computations that can be programmed without needing any data-driven learning.

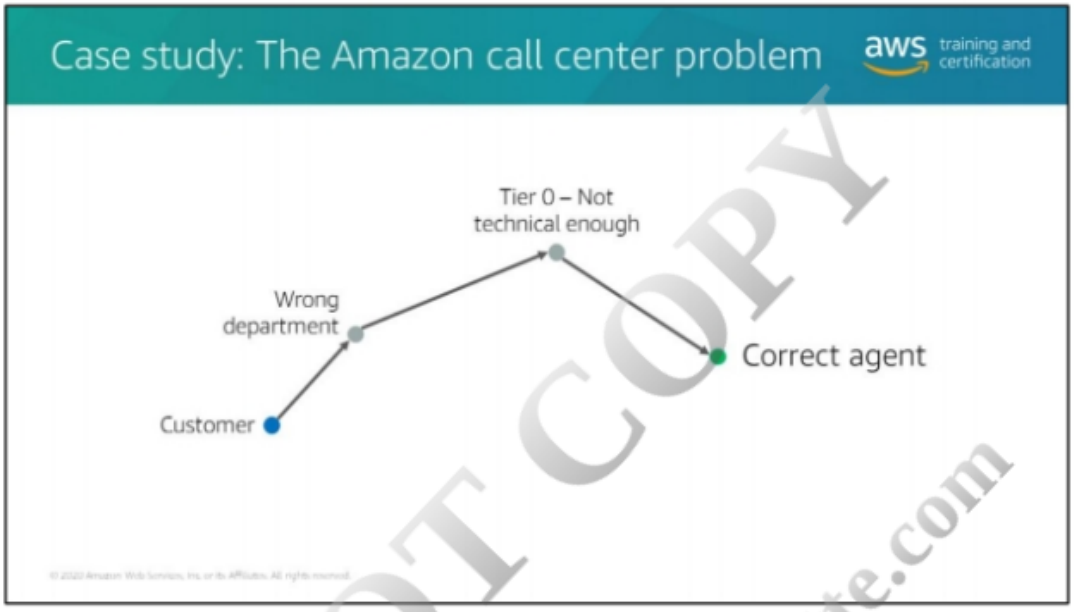
Alternatively, you should use machine learning for the following situations:

- *You cannot code the rules:* Many human tasks (such as recognizing whether an email is spam or not spam) cannot be adequately solved using a simple (deterministic), rule-based solution. A large number of factors could influence the answer. When rules depend on too many factors and many of these rules overlap or need to be tuned very finely, it soon becomes difficult for a human to accurately code the rules. You can use ML to effectively solve this problem.
- *You cannot scale:* You might be able to manually recognize a few hundred emails and decide whether they are spam or not. However, this task becomes tedious for millions of emails. ML solutions are effective at handling large-scale problems.

ML can be applied to many different use case. It can help extract hidden patterns from data, summarize data into concise descriptions, optimize an action, adapt an action, and more. The following domains make sense to apply machine learning:

- Categorization
- Predictive routing
- Fraud detection
- Personalized Advertising
- Voice assistants
- Face unlock
- Dynamic pricing
- Email filtering
- Self-driving cars

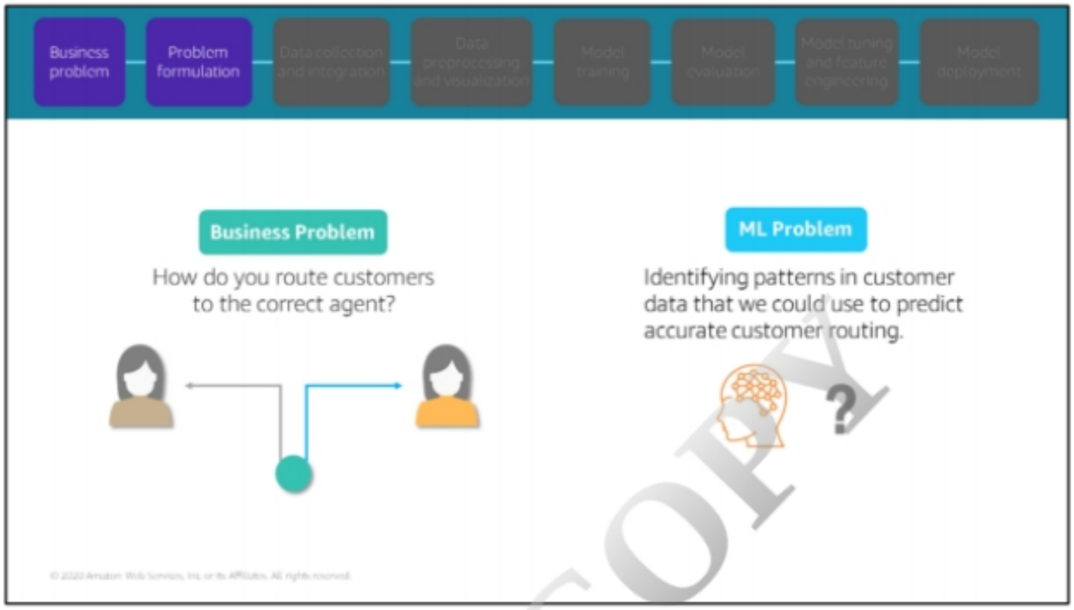
We'll dive into some of these areas a bit more later on when we introduce the various course project options, but for now - no matter what kind of machine learning application you want to use, one of the most important things you're going to need is data. The more data your machine has, like user activity, click streams, purchases, or likes, the stronger your cycle of improvement can be.



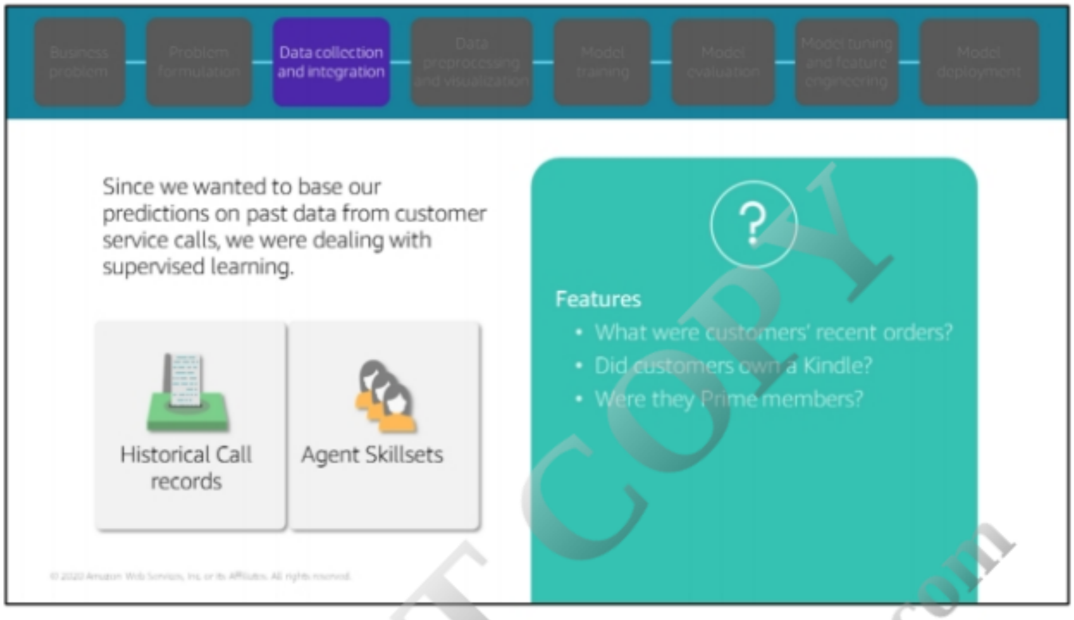
Several years ago, Amazon needed to improve the way it routed customer service calls, so it looked to machine learning for help. The original routing system worked something like this: a customer would call in and was greeted by a menu. "Press 1 for Returns. Press 2 for Kindle. Press 3 for…", well, you get the idea. The customer would then make a selection and be sent to an agent who would be trained in the right skills to help the customer.

During the problem formulation phase of the pipeline, Amazon determined that the current routing system was problematic. Thinking about Amazon, you may be able to guess that we do and sell a lot of stuff here, and so the list of things a customer might be calling us for is, well, just about endless. So if we didn't play the right option to a customer calling in, the customer might be sent to a generalist or even to the wrong specialist, who then had to figure out what the customer needed before finally sending them to the agent with the right skills.

For some businesses, maybe that's not the end of the world. For Amazon, dealing with hundreds of millions of customer calls a year, it was pretty inefficient. It cost a lot of money, wasted a lot of time, and worst of all, it was not a good way to get our customers the help they needed.

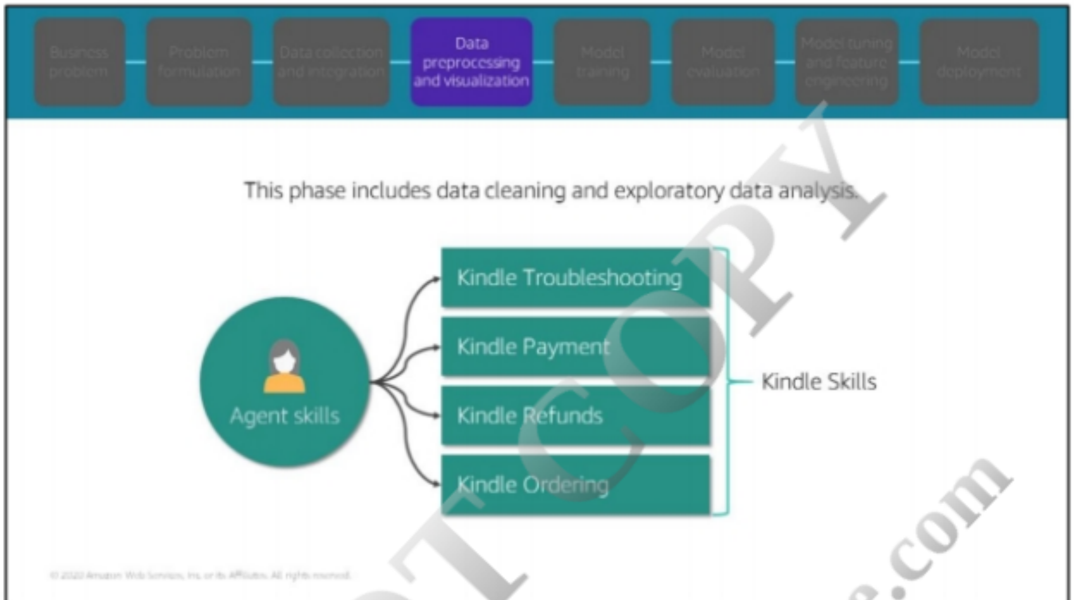


So the business problem that was articulated at this phase was focused around figuring out how to route customers to agents with the right skills and, therefore, reduce call transfers. To solve this problem, we needed to predict what skill would solve a customer call. Converting to a machine learning problem this became: identifying patterns in customer data that we could use to predict accurate customer routing. Based on the wording of this ML problem, it was clear that we were dealing with a multiclass classification problem.



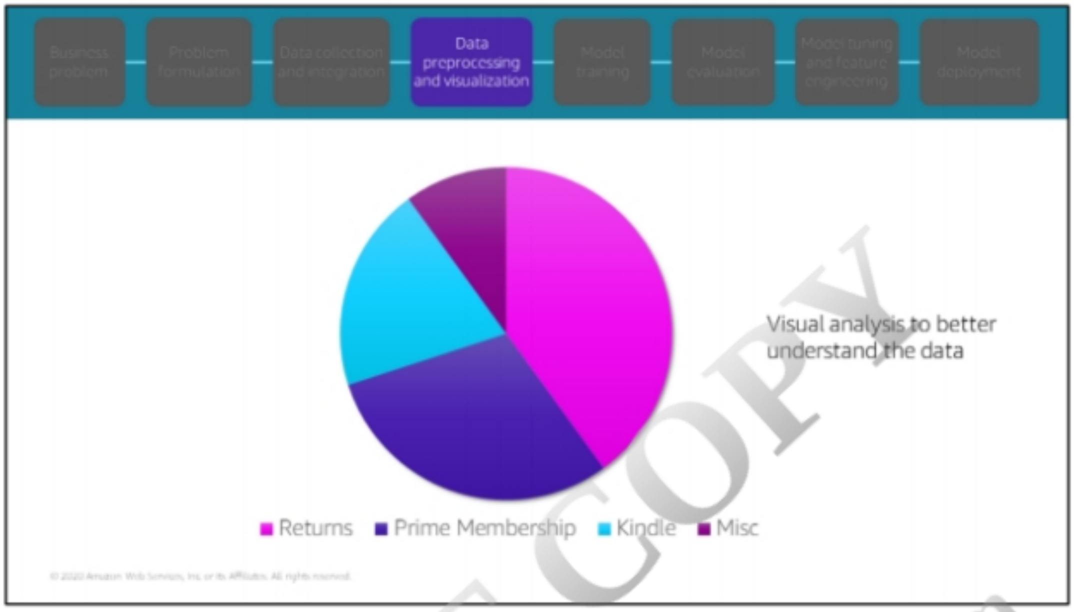
Since we wanted to base our predictions on past data from customer service calls, we were dealing with supervised learning. We eventually would train our model on a bunch of historical customer data that included the correct labels or customer agent skills. That enabled the model to make its own predictions on other similar data moving forward, predicting, say, that a customer call needed a Kindle skill.

The data we needed for this came from answering questions like what were customers' recent orders? Did customers own a Kindle? Were they Prime members? The answers to these questions became our features.

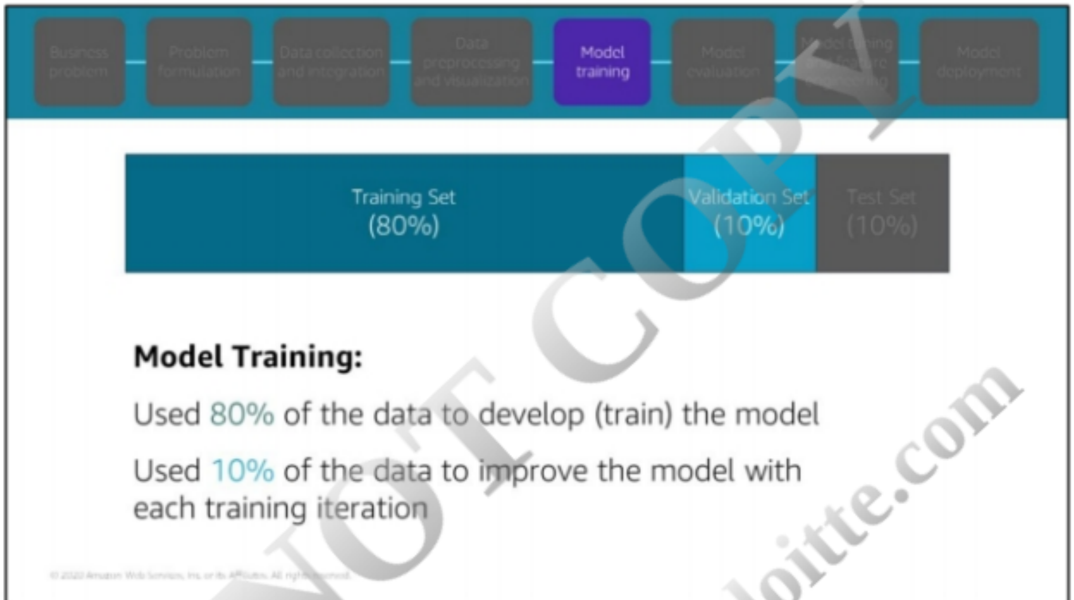


We then moved on to the data preparation or preprocessing phase. This phase, as we'll detail later, includes data cleaning and exploratory data analysis. Now a lot was done at this point, but one example of data analysis undertaken at this point was to think critically about the labels we were using. We asked ourselves, are there any labels that we want to exclude from the model for some business reason? Are there any labels that are not entirely accurate? Any labels similar enough to be combined? Finding answers to some of these questions by exploring the data would help cut down on the amount of features being used and simplify our model.

An example of what may have been found in this type of analysis was instead of having labels represent multiple Kindle skills, it made more sense to combine those skills into one overarching Kindle skill label, so that every customer who had a problem with a Kindle was routed to an agent trained in all Kindle issues.



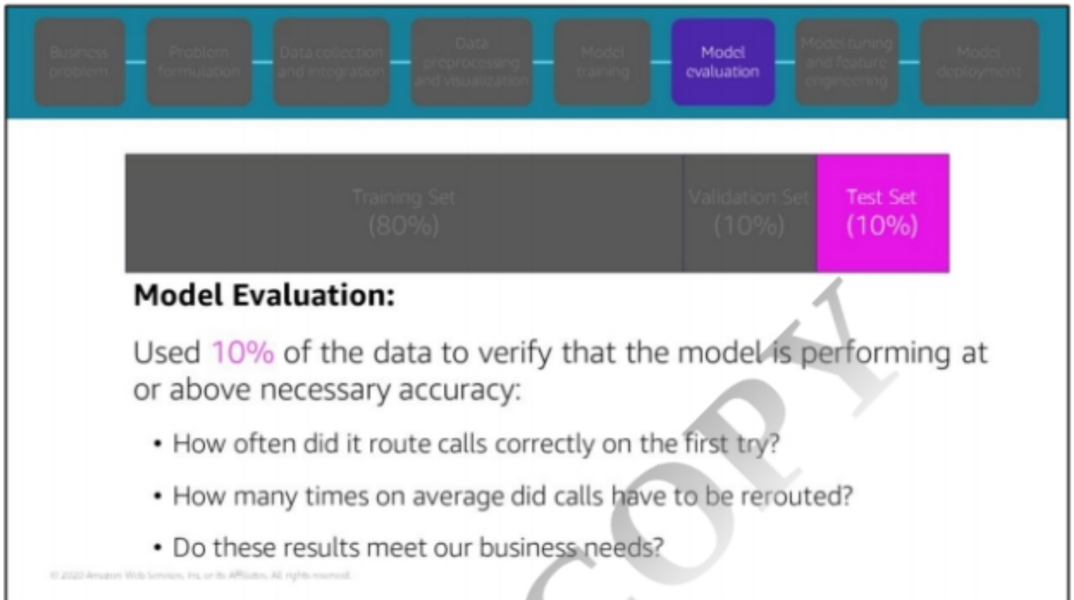
Data visualization was the next step, where we did a number of things including a programmatic analysis to give us a quick sense of feature and label summaries – effectively helping us better understand the data we were working with. At this point we may have noticed that 40% of calls were related to returns, 30% were related to Prime memberships, 30% were related to Kindle, and so on. Important information that gives us a better sense of the data we're working with.



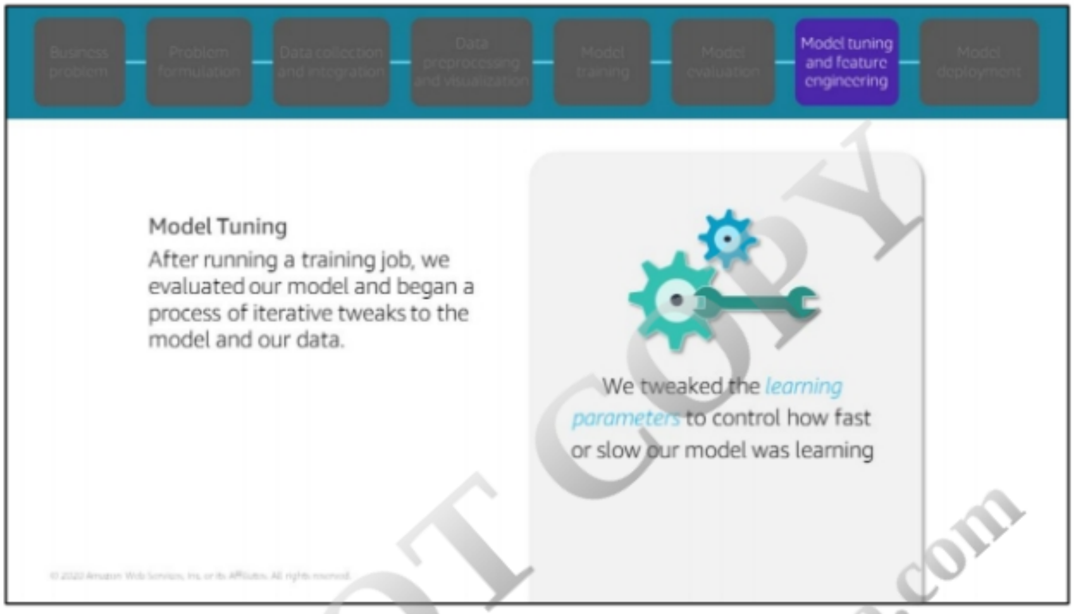
A big part of preparing for that training process is to first split your data to ensure a proper division between your training and evaluation efforts.

Think about it this way. The fundamental goal of ML is to *generalize* beyond the data instances used to train models. You want to evaluate your model to estimate the quality of its predictions for data the model has not been trained on. However, as is the case in supervised learning, because future instances have unknown target values and you cannot check the accuracy of your predictions for future instances now, you need to use some of the data that you already know the answer for as a proxy for future data. Evaluating the model with the same data that was used for training is not useful, because it rewards models that can "remember" the training data, as opposed to generalizing from it.

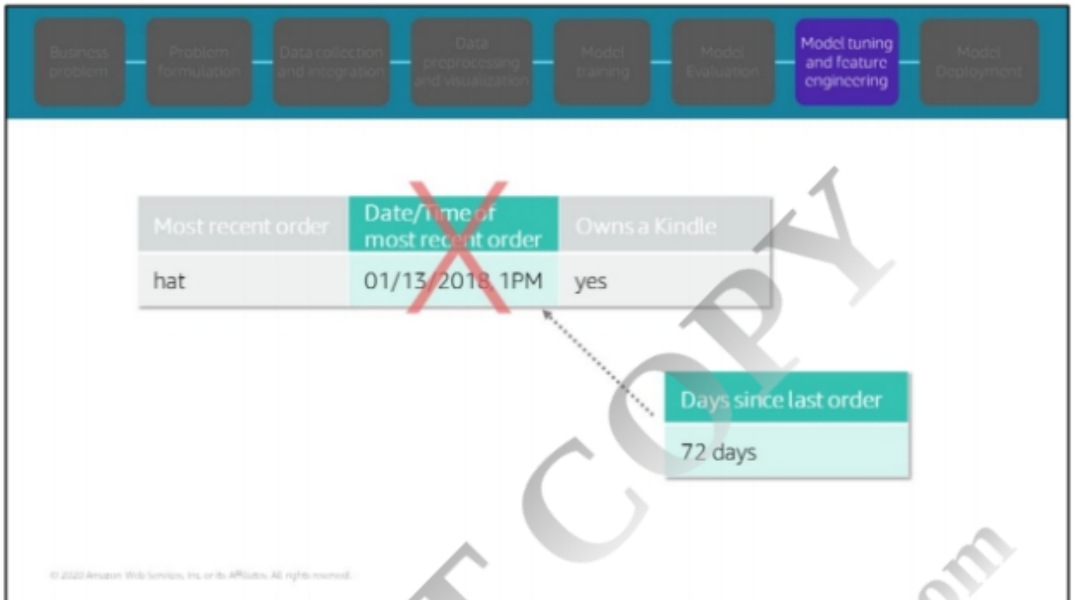
A common strategy is to split all available labeled data into training, validation, and testing subsets, usually with a ratio of 80%:10%:10%. (Another common ratio is 70%:15%:15%.)



Once happy with how the model interacted with that unseen test data, we deployed the model into production and monitored it to make sure that our business problem was indeed being addressed. Our problem was predicated on the assumption that the ability to more accurately predict skills would reduce the numbers of transfers a customer experienced – that was put to the test after we deployed – and in fact the number transfers did decrease, resulting in a much better customer experience.

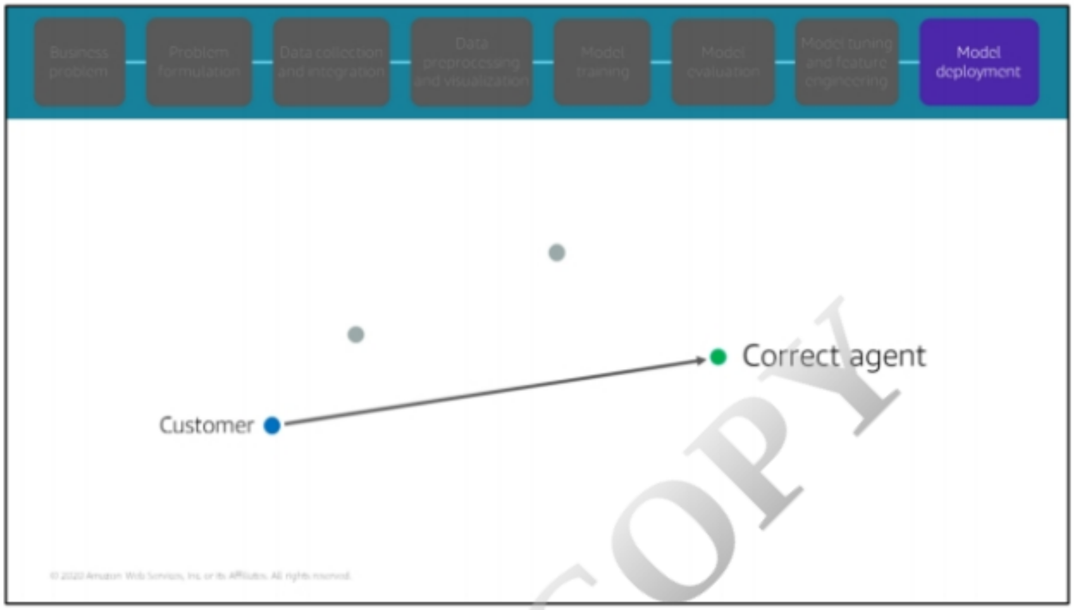


After running a training job, we evaluated our model and began a process of iterative tweaks to the model and our data. For instance, we performed hyperparameter optimization. We tweaked the *learning parameters* to control how fast or slow our model was learning. Learning too fast means that the algorithm will never reach an optimum value, while learning too slow means that the algorithm takes too long and may never converge to the optimum in the given number of steps.



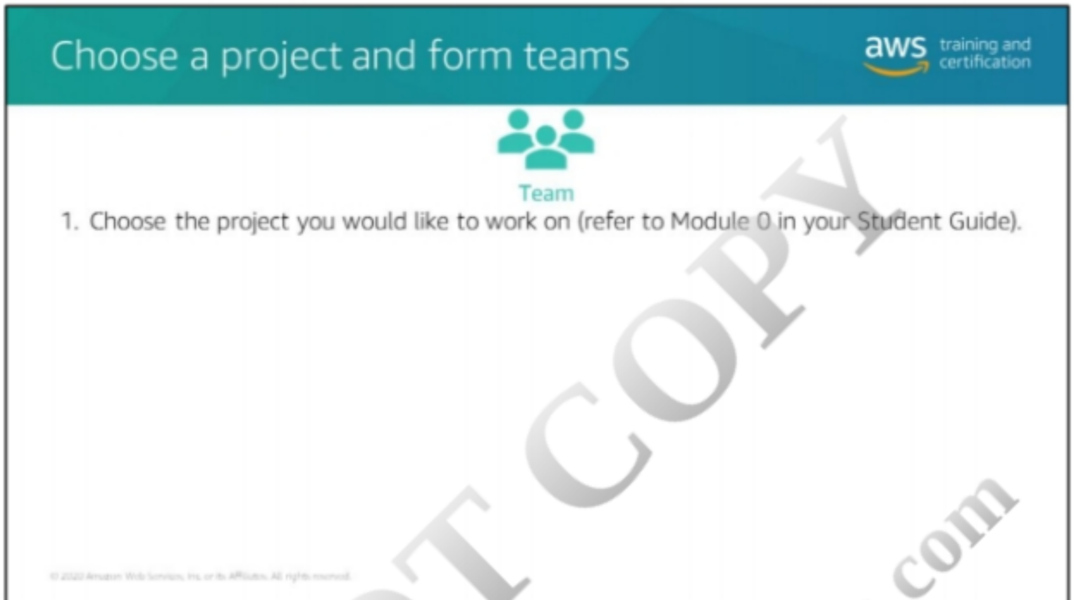
We then moved on to feature engineering. We had some features that answered questions like "What was a customer's most recent order?", "What was the time of a customer's most recent order?", and "Did the customer own a Kindle?" When we feed these features into the model training algorithm, it can only learn from exactly what we show it. Here, for instance, we are showing the model that this purchase was made at 1 PM on Tuesday the 13th. Unless we want to predict something really specific or we're doing a time series analysis, that might not be the most meaningful feature to feed into our model.

It would be more meaningful if we could transform that timestamp into a feature that represents how long ago that order took place. Knowing, for instance, that your last purchase was months ago will probably help the model realize that your last purchase probably isn't related to the reason you're calling. Obviously, we can engineer this feature by just taking the difference between the order date and time and today's date and time. Now that's a much more helpful feature.



We then deployed the model, and it now helps customers get directed to the correct agent the first time.

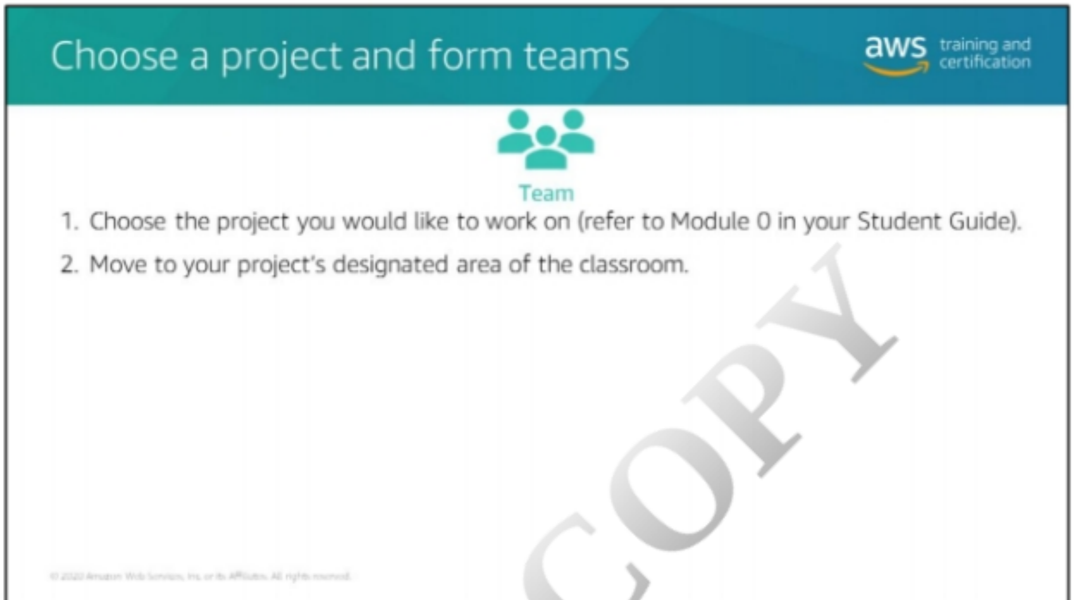
Projects



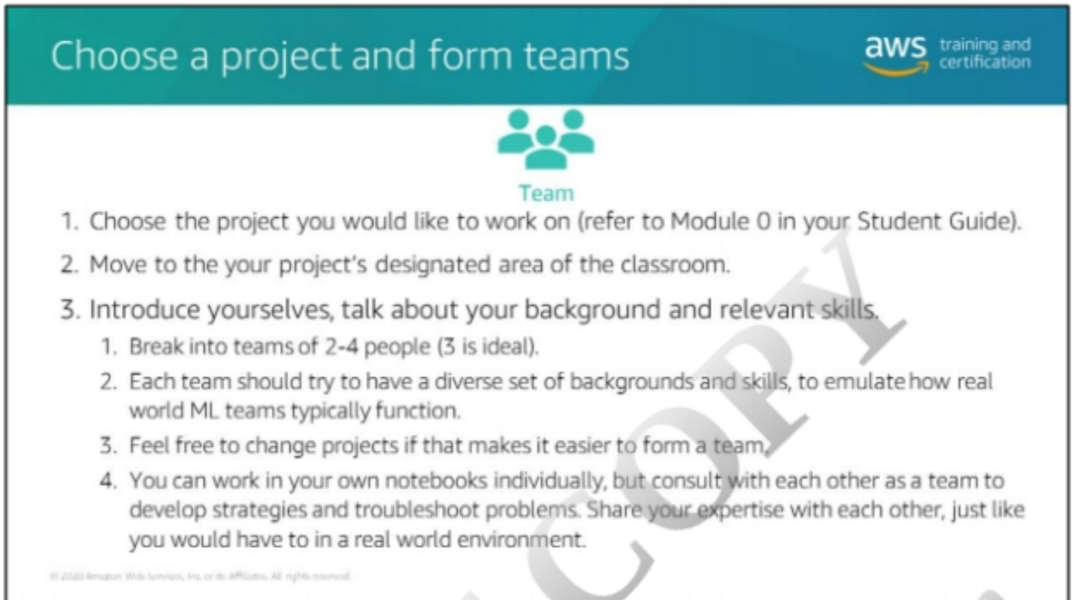
First, you'll need to choose the project you'd like to work on. If you want to re-read the materials on each project, you can go back to Module 0 in your Student Guide and the information is there for you to read through. Take a few minutes to decide.

Student project template: https://aws-tc-largeobjects.s3-us-west-2.amazonaws.com/ILT-TF-200-MLDWTS/Student+Project+Template.docx

Example project: https://aws-tc-largeobjects.s3-us-west-2.amazonaws.com/ILT-TF-200-MLDWTS/Example+Project.docx

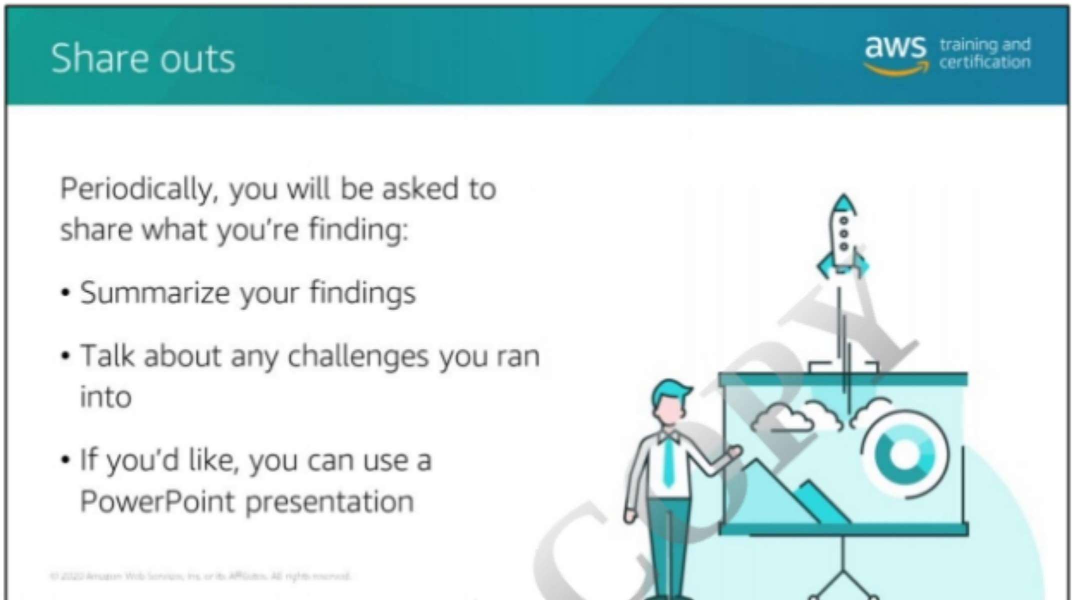


Once you've decided on your project, stand up and move to the area of your classroom that your instructor has designated for your project. This will help you form teams.



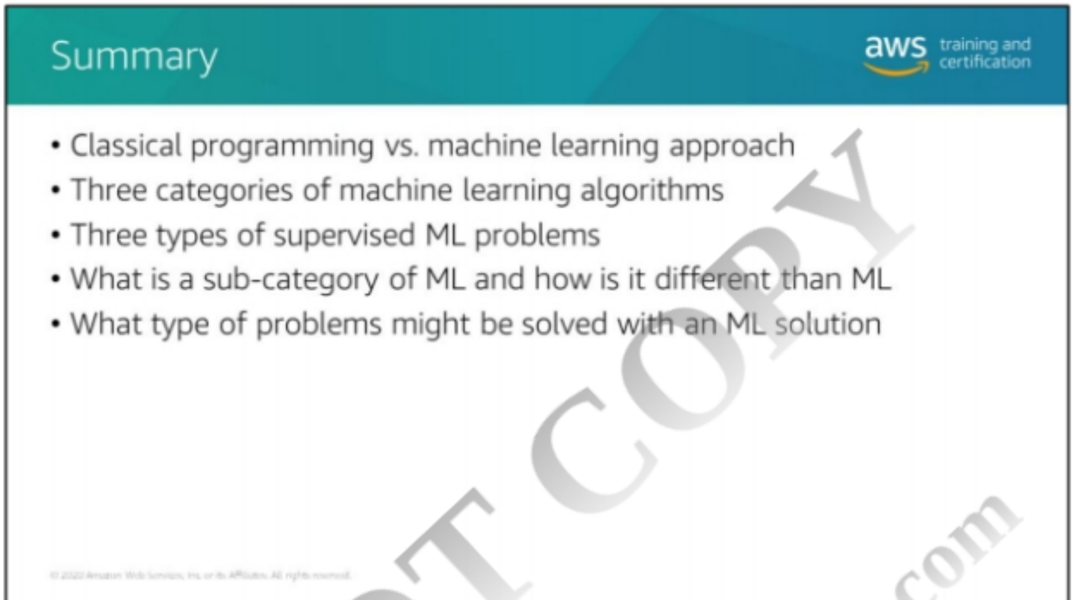
Introduce yourselves to each other, briefly talk about your background and relevant skills. This includes programming, data analysis and engineering, or anything else that might be important to know. Then break into groups of 3 (groups can be 2 or 4 if necessary, but 3 is ideal). To have the best chance at success, you should try to diversify your groups to ensure that you have a variety of skills, similar to how real world ML teams function.

If the math doesn't work out for there to be evenly distributed teams, consider changing projects to offer your skills to another team that might be in need.
Once you actually start working on your projects, you'll collaborate to develop strategies for each part of the project, but you can then work individually in your own notebook environments rather than have one person write the code for your whole group. If you are not experienced in Python, you may want to do the notebook parts with someone who is, however.



Towards the end of the class, you or your group will be asked periodically to share your findings. Above are listed the considerations you should make when sharing.

Summary



We've covered a lot in module one. You should now be able to:

Compare classical programming vs. machine learning approach

- When it comes to developing a prediction (output) from data (input), we know that there needs to be some rules applied to the data.
- In classical programming, these rules are created by humans, based on factors like business requirements and domain knowledge.
- Machine learning, by contrast, would let us use a variety of data collected in the past to automatically derive the patterns hidden in that data. The patterns are then used to create the model, which is applied to new data to provide a more well-informed and adaptive prediction.

Explain the three categories of machine learning algorithms

- Supervised, unsupervised and reinforcement

Explain the three types of supervised ML problems

- Binary classification, multiclass classification, and regression problem

Explain a sub-category of ML and how is it different than ML

- Deep learning (or DL) is a subset of machine learning. These algorithms learn using artificial neural networks

Determine what type of problems might be solved with an ML solution
- Not appropriate if you can solve it with simple rules or computations
- Is appropriate if you cannot code the rules to make a prediction or cannot scale predictions

Before we move on to the next module let's go through a few knowledge checks.

Knowledge Checks

How do machine learning approaches differ from classical programming approaches to solving problems?

Knowledge Checks

How do machine learning approaches differ from classical programming approaches to solving problems?

Classical programming:

- Requires humans to explicitly identify patterns themselves from the data.
- Does not provide generalizable solutions.

Machine learning:

- Derives patterns from data automatically
- Provides generalizable solutions.

Classical programming: It requires humans to explicitly identify and then code the patterns themselves from the data. It's not generalizable to anything other than what it was created for.

Machine learning: The models are derived from data automatically. It's generalizable: it can adapt easily to small changes in the patterns in the data.

Knowledge Checks

aws training and certification

You are given a task to identify employees who are potentially at risk for quitting, based on the historical data about the company's employees and their retention rates. Which type of machine learning would you use to address this?

Knowledge Checks

You are given a task to identify employees who are potentially at risk for quitting, based on the historical data about the company's employees and their retention rates. Which type of machine learning would you use to address this?

Supervised

Supervised, because you have the labeled historical data.

Knowledge Checks

A movie streaming company is trying to tag each movie as it's added into different genres. they have historical information about the description of the movie along with other reviewed articles. What type of ML can they use to for this problem?

A. Binary classification
B. Multi-class classification
C. Regression
D. Reinforcement Learning

Knowledge Checks

A movie streaming company is trying to tag each movie as it's added into different genres. they have historical information about the description of the movie along with other reviewed articles. What type of ML can they use to for this problem?

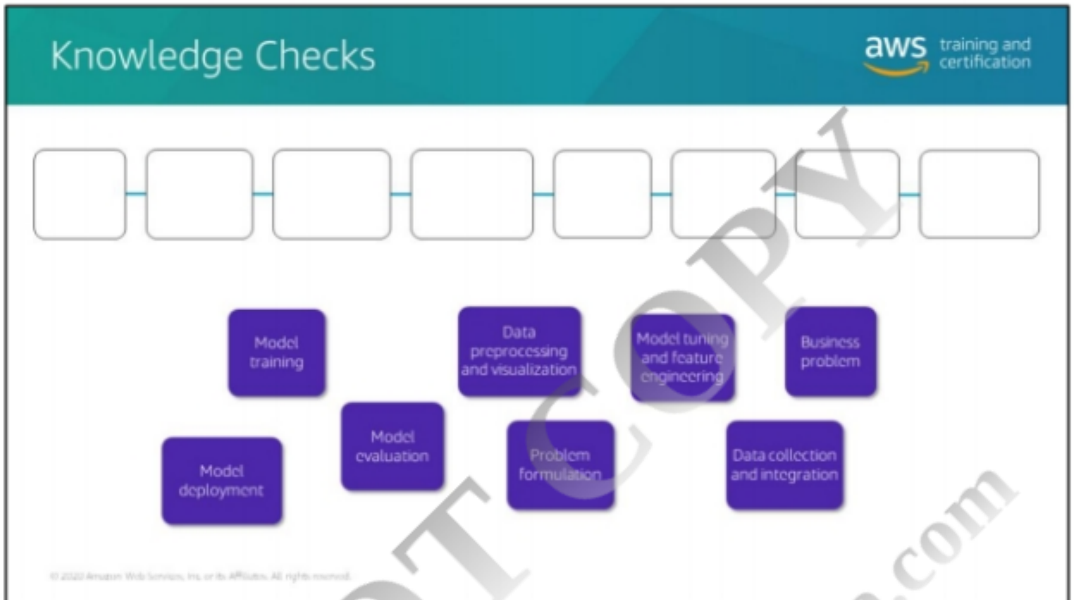
A. Binary classification
B. Multi-class classification
C. Regression
D. Reinforcement Learning

A is incorrect, because you will have more than two categories to sort the movies into.

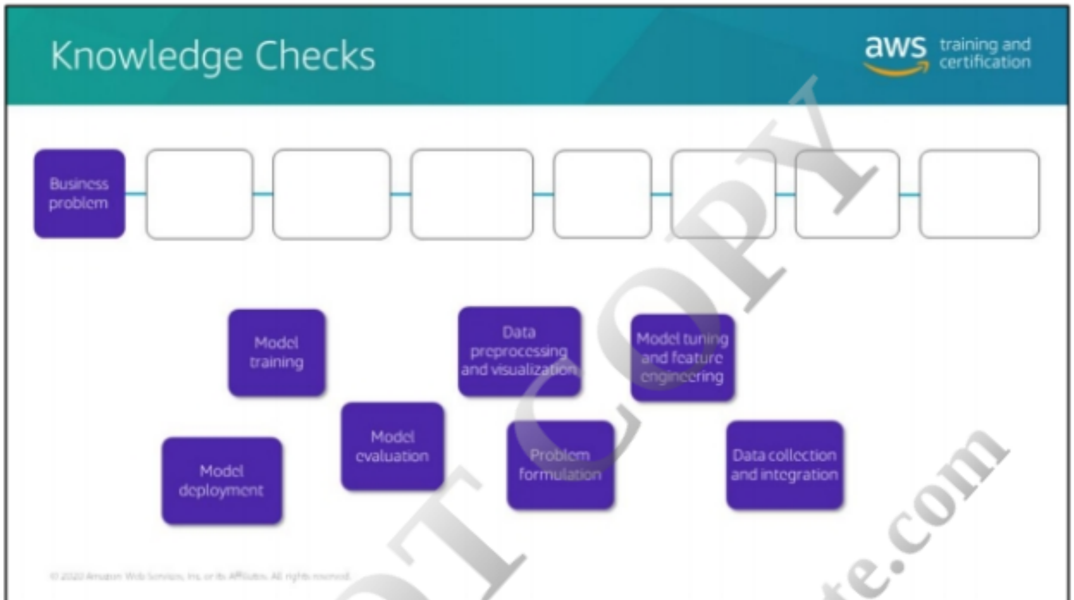
C is incorrect, because it could result in infinite number of categories, with movies sorted into very specific, less helpful groups.

D is incorrect, because you don't have an environment to learn from; you will be using historical data instead.

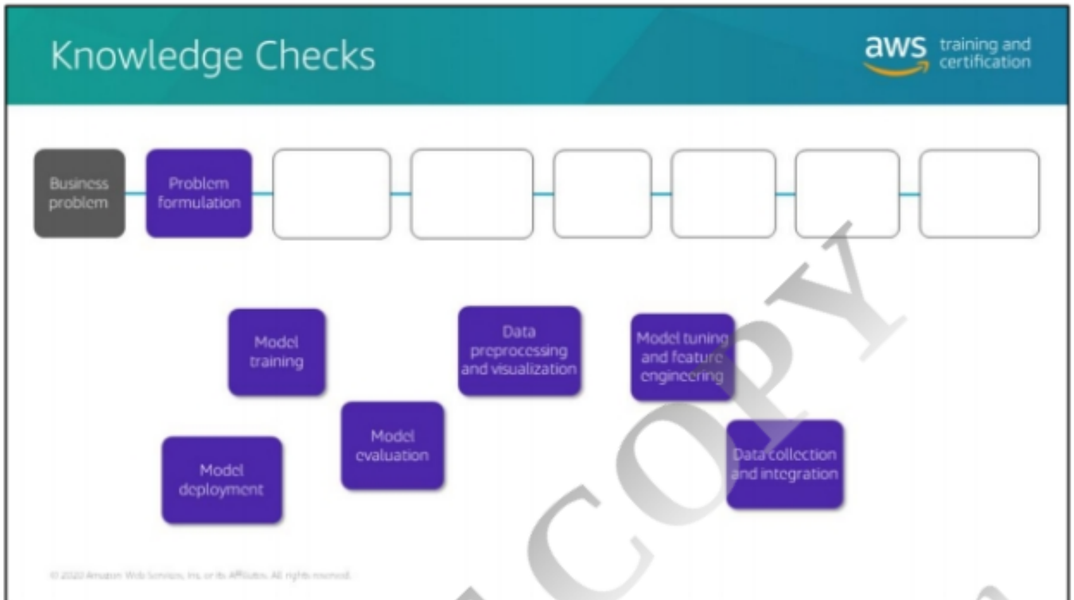
B is correct, because you will be sorting the movies into 3 or more categories.



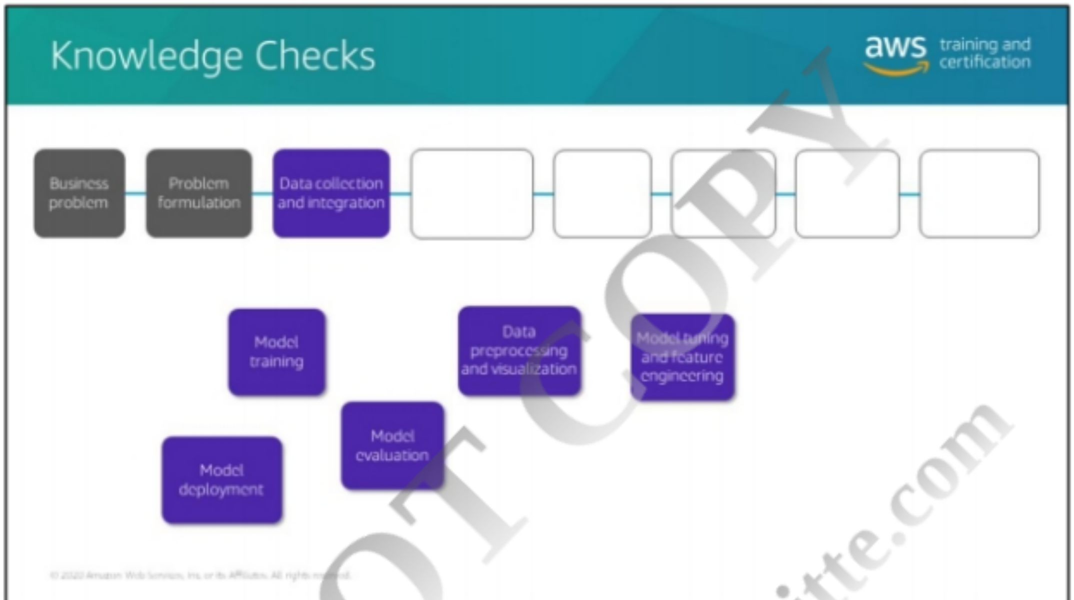
Put the boxes in the correct order to rebuild the ML Pipeline.



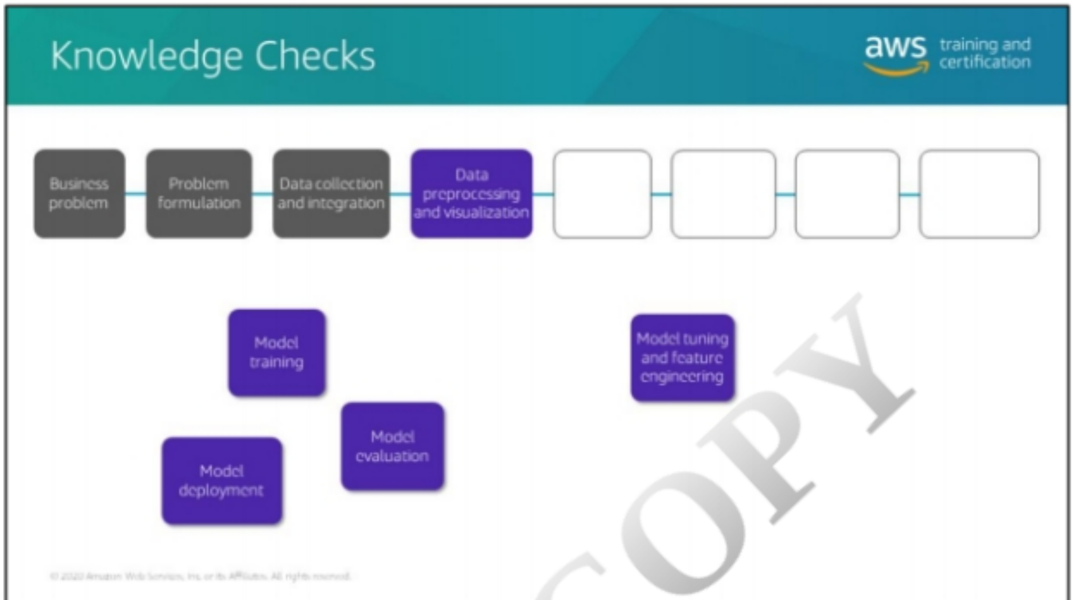
You have to start with a business problem that might be able to be addressed by ML.



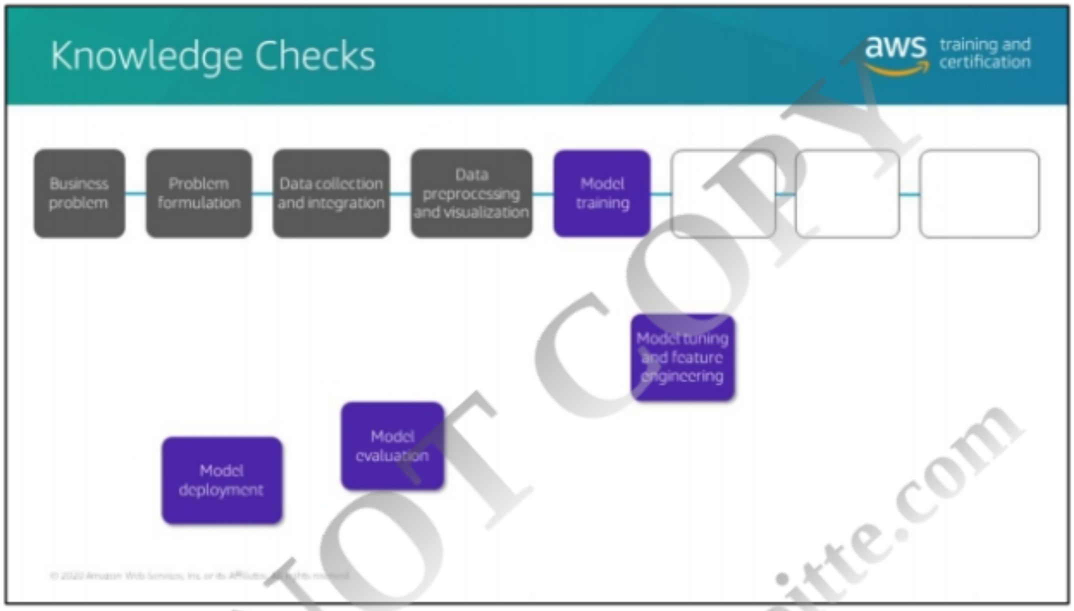
Problem formulation is where you convert the business problem into a machine learning problem.



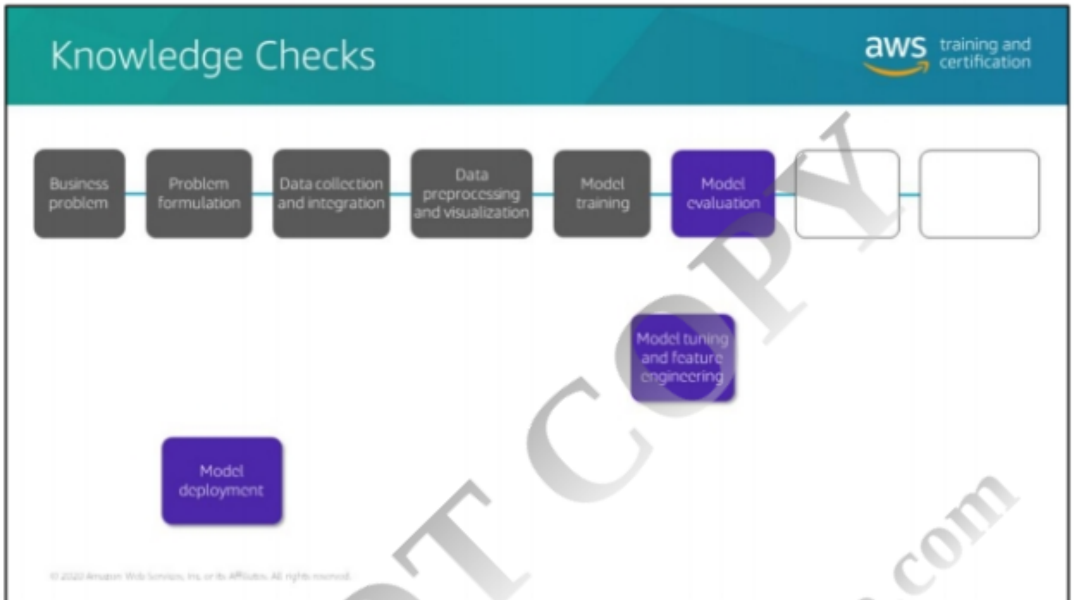
After you have your machine learning problem, you need data. Data collection and integration is where you collect and ingest the data from its original sources.



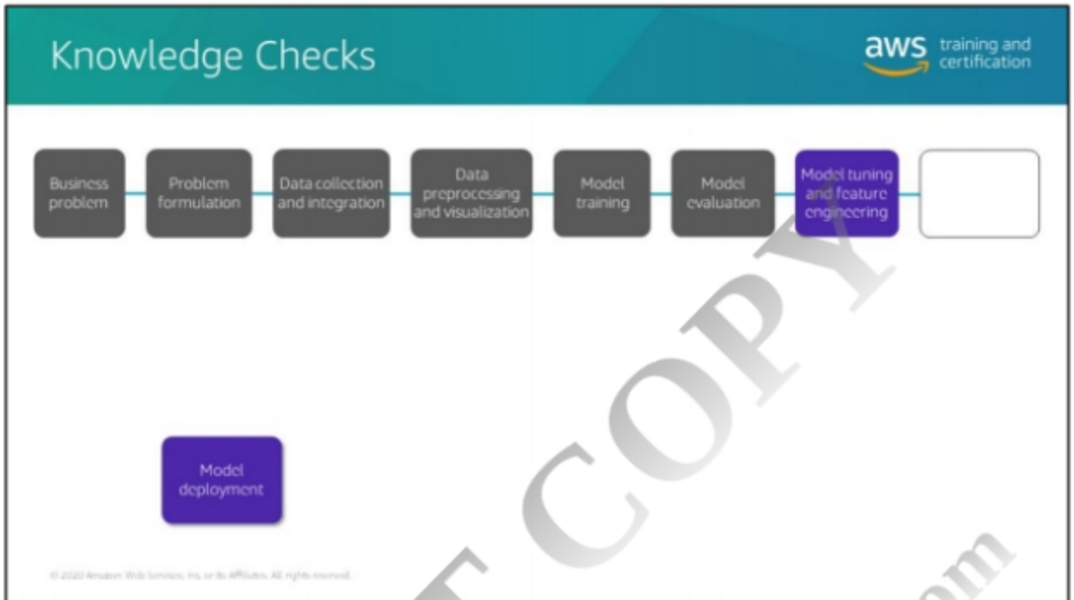
After you've collected and integrated your data, data preprocessing includes preparation where you clean and prepare the data for the next step. Additionally, visualization/analysis is where you look through the data to try to identify patterns.



Model training and tuning is next, and that's where your algorithm learns from the patterns in the data.

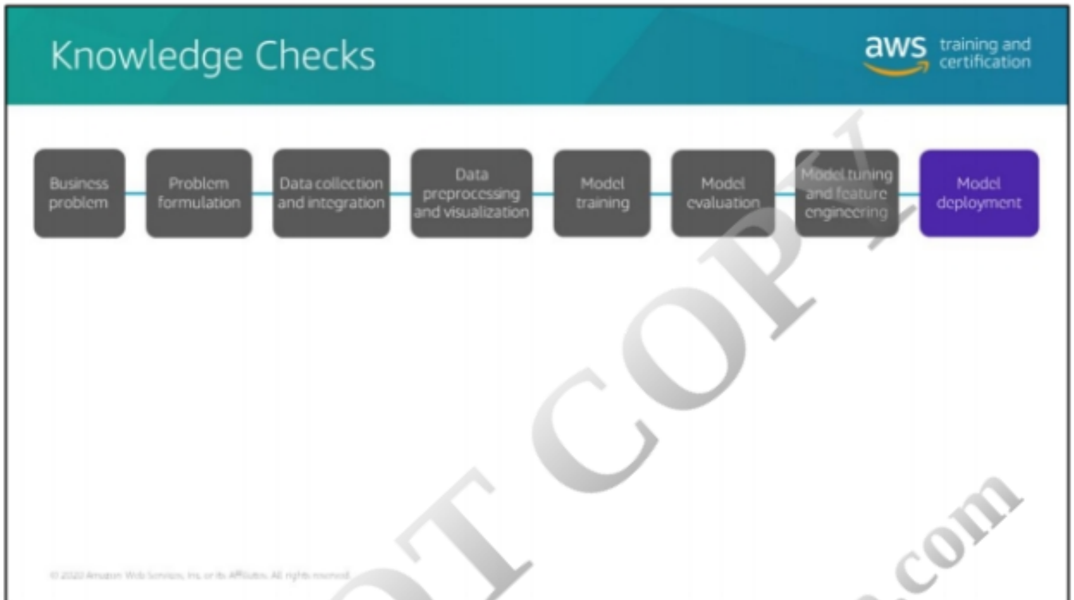


After you've trained your model you need to evaluate how it performs. Model evaluation is where you evaluate how well the model performed based on identified metrics.

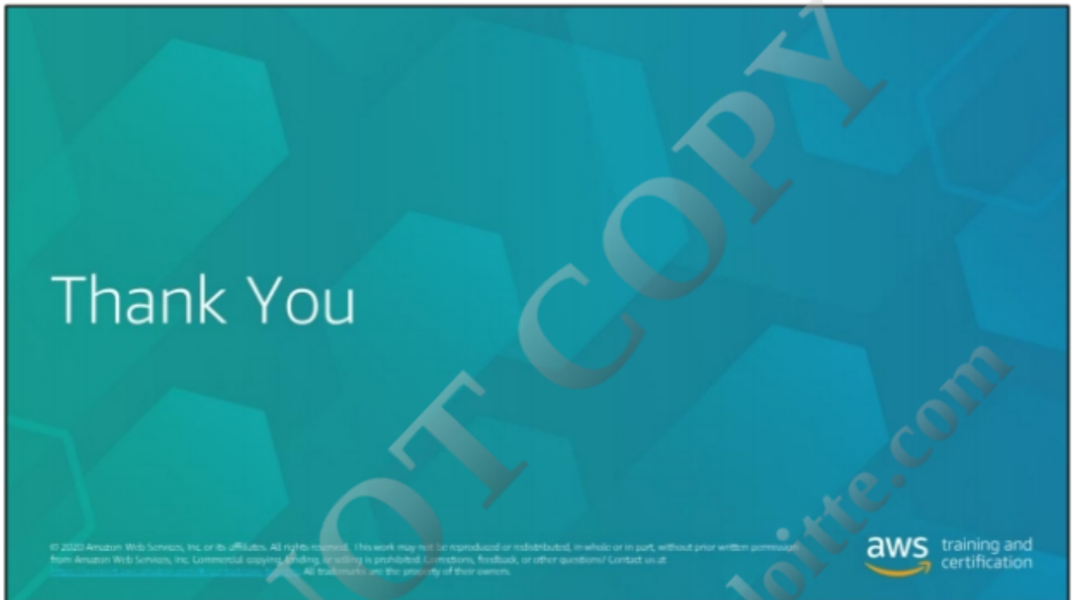


After seen how your model does, you'll want to improve it. Feature engineering is the process of selecting and improving features. Feature selection is where you identify which characteristics of the data are likely to be useful for making predictions.

Feature augmentation is how you iteratively improve the performance of your model by changing your features.



Model deployment is when you put your model into production.


Thank You
aws training and certification